Лабораторная работа по Linear-Regression

Федоров Артем Максимович

317 группа, Ноябрь 2023

1 Введение

Решение проблемы классификации объектов на конечное, заранее определенное число классов является классической задачей, решаемой методами машинного обучения. Такой класс задач подразумевает под собой нахождение алгоритма, закона, что наиболее точно определял бы зависимость между объектом с признаками и соответствующего класса.

На практике, редко удается построить такое решение, что могло бы достоверно точно определять данную зависимость, что необязательно должно следовать из неидеальности применяемых моделей, но вполне может появиться и ввиду зашумленности данных, неидеальности системы классов и простой недообученности модели по причине отсутствия достаточного размера или смещенности обучающей выборки. Потому умение модели вместо одной лишь классификации оценивать и вероятность для каждого объекта быть причисленным к тому или иному классу является крайне важной и очень удобной на практике особенностью алгоритмов. В классе линейных моделей такой характеристикой, как предсказание вероятностей, обладает Логистическая регрессия **(Ligistic-Regression)**

Одной из сфер применения машинного обучения, где повсеместно возникают задачи классификации, является раздел задач распознавания естественного языка (**NLP** - **N**atural **L**anguage **P**roccessing), в рамках возможных проблем которого могут встречаться задачи распознавания тематик научных документов, ценностной направленности текстов, поляризации мнений авторов, определения эмоциональной окраски текстов и так далее.

Данный отчет стремится подвести итог экспериментам, проводимых в рамках изучения Логистической регрессии и ее работы в рамках задачи по категоризации человеческой речи с применением градиентных методов оптимизации.

2 Начальные данные

Задачи **NLP**, как задачи построения интерфейса взаимодействия человека с компьютером, сильно зависят от представления первоначальных данных в обрабатываемый компьютером код. Потому рассматриваемая задача бинарной классификации текстов и ее итоговое решение нераздельно связаны с обработкой классифицируемых объектов. Важной задачей является правильно интерпретировать данные и построить на них оптимальное решение.

2.1 Данные задачи

В задании рассматривается набор текстовых комментариев, собранных в интернете и размеченных на два класса соответственно: "комментарий не токсичный" и "комментарий токсичный". В дальнейшем множеством классов обозначается $\mathcal{Y} = \{-1, 1\}$, где положительный класс — обозначение токсичности комментария. Тогда естественно построить объект $\mathfrak{U}^{æ}$ как множество любых символьных цепочек произвольной длины меньшей чем константа æ (условие на конечность), в дальнейшем именуемых **текстами**, приходящими на вход алгоритму. Пусть $\hat{\mathfrak{N}}^{æ}$ есть конечное множество объектов из $\mathfrak{U}^{æ}$, для которого определен оператор перевода $\mathcal{V} : \hat{\mathfrak{N}}^{æ} \to \mathcal{X} \equiv \mathbb{R}^{q \times l}$, осуществляющий отображение каждого объекта-строки из множества мощности l в пространство вещественных векторов размерности q, где $l = |\hat{\mathfrak{N}}^{æ}|$. Тем самым получен способ построения эмбеддингов (**embedding**) в признаковом пространстве $\mathbb{R}^q$.

2.2 Теоретический фундамент

После построения оператора перевода и получения эмбеддингов текстов, задача становится решаемой стандартными средствами машинного обучения. Будем считать, что такое представление

верно и отражает каждый из объектов наиболее точно, а потому от представления начальных текстовых комментариев как $u_i \in \mathfrak{U}^{\text{æ}}$ перейдем к преставлению этих объектов их эмбедингами из $\mathcal{X}$. На декартовом произведении $[\mathbb{R}^q, \mathcal{Y}]$ определена операция бинарного соответствия $\sim$: будем считать, что $x \sim y$, где $x \in \mathcal{X}$ и $y \in \mathcal{Y}$ тогда и только тогда, когда объекту x поставлен в соответсвие класс y. Тогда построим стохастическую модель задачи, при которой каждый объект порождает дискретное вероятностное распределение на множестве $\mathcal{Y}$ как:

$x \sim y$	$x \sim -1$	$x \sim 1$
$P(x \sim y)$	$P(x \sim -1)$	$P(x \sim 1)$

Такое распределение является Бернулиевским. Пусть X^l — выборка эмбеддингов мощности l из пространства $\mathcal{X}$. Для такого распределения построим оценку максимального правдоподобия по выборке X^l с соответственным вектором $y = \{y_i : x_i \sim y_i\}$ и вектором параметров $w \in \mathbb{R}^{q+1}$ и проведем оптимизацию по параметру w

> Здесь и далее для простоты выкладок будем считать вектор $w \in \mathbb{R}^{q+1}$, как вектор, состоящий из двух частей:
>
> - $\hat{w} \in \mathbb{R}^q$ вектор весов соответственных признаков
> - $w_0 \in \mathbb{R}$ bias линейной регрессии

$$\hat{L}(X^l, y; w) = \prod_{i=1}^{l} (P(x_i \sim -1|w))^{[y_i=-1]} (P(x_i \sim 1|w))^{[y_i=1]} \longrightarrow \max_{w \in \mathbb{R}^{q+1}}$$

Такая постановка оптимизационной задачи ввиду монотонности функции логарифма эквивалентна:

$$L(X^l, y; w) = -\frac{1}{l} \sum_{i=1}^{l} [y_i = -1] \ln P(x_i \sim -1|w) + [y_i = 1] \ln P(x_i \sim 1|w) \longrightarrow \min_{w \in \mathbb{R}^{q+1}}$$

Остается вопрос о том, как высчитывать вероятность каждого класса быть причисленным соответственному объекту при фиксированном w. Выбираем приближение вероятности функцией сигмоидой: $P(x_i \sim 1) = \sigma(x_i, w) = \sigma(M_i) = \frac{1}{1+e^{-M_i}}$, где $M_i = \langle \hat{w}, x_i \rangle - w_0$. Алгоритм, решающий поставленную задачу, называется Логистической регрессией, а саму задачу оптимизации, приняв во внимание соотношение $1 - \sigma(M_i) = \sigma(-M_i)$, можно переписать в виде:

$$L(X^l, y; w) = \frac{1}{l} \sum_{i=1}^{l} \ln(1 + exp\{-y_i \langle w, x_i \rangle - w_0\}) \longrightarrow \min_{w \in \mathbb{R}^{q+1}}$$

2.2.1 Нахождение градиента

Для реализации градиентного алгоритма численного решения поставленной оптимизационной задачи требуется знание производной от функции эмпирического риска $L(X^l, y; w)$.

$$d(L(X^l, y; w)) = \frac{1}{l} \sum_{i=1}^{l} d(\ln(1 + exp\{-y_i \langle \hat{w}, x_i \rangle - w_0\}))$$

$$d(L(X^l, y; w)) = -\frac{1}{l} \sum_{i=1}^{l} \frac{e^{-M_i}}{1 + e^{-M_i}} d(y_i \langle \hat{w}, x_i \rangle - w_0)$$

$$d(L(X^l, y; w)) = -\frac{1}{l} \sum_{i=1}^{l} \frac{e^{-M_i}}{1 + e^{-M_i}} y_i \langle d\hat{w}, x_i \rangle$$

$$d(L(X^l, y; w)) = \left\langle dw, -\frac{1}{l} \sum_{i=1}^{l} \frac{e^{-M_i}}{1 + e^{-M_i}} y_i x_i \right\rangle$$

Из чего следует явный вид градиента функции:

$$\Longrightarrow \nabla L(X^l, y; w) = -\frac{1}{l} \sum_{i=1}^{l} \frac{e^{-M_i}}{1 + e^{-M_i}} y_i x_i$$

2.2.2 Мультиномиальная логистическая регрессия

Пусть теперь стоит задача мультиклассовой классификации. В таком случае множество классов задачи представимо в виде $\tilde{\mathcal{Y}} = \{1, 2, 3, ..., m\}$. В соответствии с бинарной задачей классификации, логично положить стохастическую модель в которой бинарное отношение $\sim$ определено на декартовом произведении $[\mathcal{X} \times \hat{\mathcal{Y}}]$, а каждый объект из $\mathcal{X}$ порождает дискретное вероятностное распределение вида:

$x \sim y$	$x \sim 1$	...	$x \sim m$
$P(x \sim y)$	$P(x \sim -1)$	...	$P(x \sim m)$

Оценка максимального правдоподобия и соответсвующая задачи оптимизации примут вид:

$$\hat{L}(X^l, y; W) = \prod_{i=1}^{l} \prod_{j=1}^{m} (P(x_i \sim j | w_j))^{[y_i = j]} \longrightarrow \max_{w \in \mathbb{R}^{q+1 \times m}}$$

> Заметим, что в общем случае веса для каждого из классов нельзя считать одинаковыми, а потому рассматривается не вектор весов, а матрица $W \in \mathbb{R}^{m \times (q+1)}$, из чего выходит, что функция $\sigma(x_i; w_j)$ больше не может считаться функцией оценки вероятности, так как не выполняется условие нормировки. Для этого используется подход **Soft-Max**, для которого вероятность каждого класса для объекта представима в виде: $P(x \sim j) = \frac{e^{z_j}}{\sum_{k=1}^{m} e^{z_k}}$, где $z_i = \langle w_j, x \rangle + w_{0j}$

Запишем классическое представление оптимизационной задачи для мультиномиальной логистической регрессии:

$$L(X^l, y; W) = -\frac{1}{l} \sum_{i=1}^{l} \sum_{j=1}^{k} [y_i = j] \ln \left(1 + \exp\left(- \langle w_j, x_i \rangle\right)\right)$$

$$\nabla_{w_k} L(X^l, y; W) = \frac{1}{l} \sum_{i=1}^{l} [y_i = k] \frac{\exp\left(- \langle w_k, x_i \rangle\right)}{1 + \exp\left(- \langle w_k, x_i \rangle\right)} x_i = \frac{1}{l} \sum_{i=1}^{l} [y_i = k] \left(1 - \sigma\left(\langle w_k, x_i \rangle\right)\right) x_i$$

2.2.3 Сведение мультиномиальной Логистической регрессии к бинарной

Пусть $m = 2$, $\mathcal{Y} = \{1, 2\}$. Таким образом задача вновь сводится к максимизации оценки максимального правдоподобия:

$$\hat{L}(X^l, y; w) = \prod_{i=1}^{l} \prod_{j=1}^{2} (P(x_i \sim j | w_j))^{[y_i = j]} = \prod_{i=1}^{l} P(x_i \sim 1 | w_1))^{[y_i = 1]} P(x_i \sim 2 | w_2))^{[y_i = 2]} \longrightarrow \max_{W \in \mathbb{R}^{q+1 \times 2}}$$

Положив, $z_{ij} =< w_j, x_i > + w_{0j}$ получим запись для оценки вероятности первого класса для i объекта:

$$P(x_i \sim 1 | W) = \frac{e^{z_{i1}}}{e^{z_{i1}} + e^{z_{i2}}} = 1 - \frac{e^{z_{i2}}}{e^{z_{i1}} + e^{z_{i2}}} = 1 - P(x_i \sim 2 | W)$$

Из чего следует, что вероятность второго класса напрямую выражается из вероятности первого. Тем самым выполняется не только условие нормировки вероятностей, но и совпадение постановки задачи мультиномиальной логистической регрессии с числом классов равным $m = 2$ с постановкой таковой для бинарной логистической регрессии.

3 Составление признакового пространства

Важной составляющей решения является определение оператора перевода строковых данных в числовые векторы признакового пространства. В данной работе принято решение воспользоваться методом построения мешка слов (**BagOfWords**). Такой метод предполагает, что исходный смысл каждого из предложений сокрыт в наборе слов, в нем используемых. Таким образом мы выделяем новое понятие — слово — как смысловую единицу, отказываясь от упорядоченности слов и разбиения текстов на предложения, абзацы и так далее; само же действие оператора перевода на корпус текстов осуществляется примерно следующим образом:

1. Пройдя по всему корпусу текстов, собрать данные о всех словах: число различных слов, число встреч каждого слова, возможно составления набора подцепочек слов и так далее.

2. За второй проход по корпусу, каждому предложению в соответствие поставить вектор, размерности числа различных слов, встреченных за первый проход, где каждому слову ставится в соответствие некоторое число, полученное как функция от числа встреч этого слова в данном тексте от данных, полученных за первый проход.

И уже в определении самой функции сопоставления слову числа в получаемых векторах эмбедингов является главной задачей в построении нашего оператора перевода $\mathcal{V}$

3.1 Предобработка текстов

Исходя из полученного определения оператора перевода текстовой информации в удобную для компьютера векторную, получаем необходимость в определении понятия слова и его информативности. Последнее желание понятно, так как отсутствие влияния слова на смысловую окраску предложения сигнализирует, что такое слово, отражаясь на конечном признаковом пространстве, будет лишь мешать работе модели. Потому введем требования на лексемы, что мы будем оставлять в для алгоритма сбора мешка слов:

- Лексемы (слова) разделены пробелом.
- Лексемы могут быть произвольной длины и необязательно значить что-либо на исходном языке.
- Лексемы должны быть качественными индикаторами того, что данное предложение относится к соответственному классу (присутствие данного слова в предложении дает нам возможность предположить окраску предложения).

Исходя из данного описания можно вывести конкретный список действий, применение которых могло бы улучшить качество классификации:

1. Приведение слов к нижнему регистру — так теряется смысл заглавных букв, однако слова по типу "Береза" и "березыа" обретают одинаковый смысл, что в большинстве случаев подпадает под наши нужды.

2. Замена всех специальных символов на пробелы — действие призвано убрать незначащие для классификации эмоциональной окраски текстов повторяющиеся пробелы и символы типа "<", ">", ":", "+" и так далее.

3. Замена чисел, записанных символами цифр, эквивалентными словесными записями — такой подход позволит рассматривать сложные числительные как набор из одинаковых слов, при этом делая их эквивалентными числительными, изначально записанным в словесной форме.

4. Удаление стоп-слов — удаление повсеместно используемых слов союзов, местоимений, слов паразитов и так далее, что не присущи какой-то одной эмоциональной окраске.

5. Применение Стемминга или Лемматизации — применение средств, позволяющих привести различные формы слов к одной, тем самым сильно сократив размерность мешка слов. (исследуются далее)

3.2 Перевод корпусов текстов в матричное представление

> Требуется получить представление вектора размерности l (размерности переводимой выборки), компонентами которого являются строки, в матрицу размерности $q \times l$, где q является размерностью конечного признакового пространства, а l - количество объектов в выборке соответственно.
>
> $$u^l = (u_1, u_2, ..., u_l), u_i \in \mathfrak{U}^{æ} \Longrightarrow \begin{pmatrix} x_{11} & ... & x_{1l} \\ x_{q1} & ... & x_{ql} \end{pmatrix} \in \mathbb{R}^{q \times l}$$

При определении оператора перевода $\mathcal{V}$ можно рассматривать различные эвристики, связанные с получением итогового числового коэффициента каждого признака в эмбединге. В исследовании использовались два типа операторов перевода:

1. Оператор, не учитывающий относительную частоту появления слова в корпусе ($\mathcal{V}_{cv}$) — Такой оператор дважды проходится по всему корпусу текстов, составляя множество всех слов, используемых в текстах, после чего создает матрицу встречаемости каждого из слов в каждом тексте. Тем самым эмбединг строки представим последовательностью чисел, равных числу встречаемости слова в соответствующей строке.

2. Оператор, это число учитывающий ($\mathcal{V}_{fidv}$) — действует подобно предыдущему, с той лишь разницей, что он строит не матрицу встречаемости слов, а матрицу веса каждого слова для в текстах, учитывая важность слова на основе частоты его встречаемости как тексте, так и во всем корпусе целиком.

4 Анализ градиента и его численного приближения

Задача классификации, изучаемая в данном исследовании, представляет собой бинарную задачу классификации на положительный и отрицательный класс. Описанный градиент, как формула, был получен аналитическим путем, однако существуют различные численные методы приближения производной функции.

Рассматривается приближение градиента функции эмпирического риска $L(X, y; w)$ в точке по правилу правой производной покомпонентно. Пусть σ_i^j — символ Кронекера, равняющийся единице при $i = j$, и нулю при любых других случаях. Тогда запись $\{\sigma_i^j\}$ будет обозначать вектор, где на i месте стоит 1 и 0 на всех остальных:

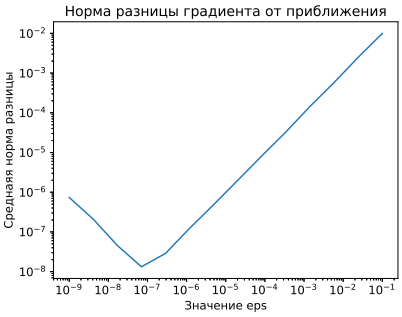

Figure 1: График зависимости нормы разницы векторов аналитического градиента и его приближения от ε

$$\nabla L(X, y; w) = \lim_{\varepsilon \to 0+0} \frac{L(X, y; w + \varepsilon h) - L(X, y; w)}{||\varepsilon h||}$$

$$\nabla_i L(X, y; w) = \lim_{\varepsilon \to 0+0} \frac{L(X, y; w + \varepsilon\{\sigma_i^j\}) - L(X, y; w)}{\varepsilon}$$

Тем самым каждая компонента вектора градиента функции эмпирического риска может быть получена как конечная разностная дробь, что из условия всюду непрерывной дифференцируемости функционала ошибки L по w при стремлении ε к нулю должна стремится к аналитической формуле градиента.

Проверка такого свойства возможна статистическим путем. Будем смотреть на убывающую в геометрической прогрессии последовательность ε_i, для каждой из которых будем генерировать 100 случайных выборок размерности 5000×32, где 32 - число признаков каждого из 5000 объектов и для каждой из них будем смотреть разницу в случайной точке w_{ij}, взятой из нормального распределения. По каждой итерации берем нормы разницы между векторами, получаемыми в одной точке с помощью аналитической формулы и формулы конечно-разностных приращений, и по каждой ε_i усредняем средним арифметическим.

На графике 1 изображена тенденция к постоянному падению средней нормы разницы между описанными векторами вплоть до определенной границы возле $\varepsilon = 10^{-7}$, после чего график средней ошибки увеличивается практически с такой же скоростью, что и снижался до этого момента. Такое наблюдение можно связать с неидеальностью представления чисел с плавающей точкой нашими компьютерами, из чего следует неэффективность использования конечно-разностных схем при точной работе, так как довольно быстро метод достигает максимума своей точности, и дальнейшее уменьшение коэффициента ε только ухудшает конечную оценку аналитической функции.

5 Исследовани градиентного спуска Логистической регрессии

Очистим изначальную выборку от специальных символов и приведем все знаки к нижнему регистру. Применим к получившемуся датасету оператор $\mathcal{V}_{cv}$, отбросив все слова, встречающиеся реже чем в 50 текстах.

Реализация решения оптимизационной задачи, поставленной во втором разделе исследования, проводится на основе градиентного спуска, при котором на каждом шагу (итерации) производится

изменение весов w в сторону антиградиента с некоторым коэффициентом. Описать структуру подхода можно следующим образом:

1. На 0 шаге алгоритма инициализируем каким-либо способом веса и строим последовательность $\{lr_i\}_{i=1}^{\infty}$, так называемую последовательность **learning-rate**, в которой $\forall i \geqslant 0 : lr_i > 0$.
2. На i шаге высчитываем градиент функции эмпирического риска gr_i при параметрах, на данный момент известных оптимизатору (X^l, y, w_i). Важным замечанием здесь является то, что градиент высчитывается сразу по **всем** объектам обучающей выборки.
3. Производится шаг градиентного спуска, при котором высчитываются новые веса алгоритма: $w_{i+1} = -lr_i \cdot gr_i$
4. Алгоритм проверяет условия выхода из цикла: например по достижению максимального числа итераций либо при достижении алгоритмом состояния, когда при i шаге функция потерь убывает не более чем на заранее заданное число

> Важным замечанием здесь является условие на определение **learning-rate**. Из теоремы о сходимости численного метода оптимизации следуют ограничения на данный ряд: $\sum_{i=1}^{\infty} lr_i$ — расходится, $\sum_{i=1}^{\infty} lr_i^2$ — сходится при таком задании последовательности можно утверждать, что алгоритм всегда сойдется к оптимуму при бесконечном количестве итераций.

Таким образом решение оптимизационной задачи и конечная модель зависит от выбора **learning-rate** и начального приближения. При чем изменение каждого из параметров неизбежно повлечет за собой изменения не только самой сходимости модели, но и ее работы на тестовой и обучающей выборках. Оценка же алгоритмов должна основываться не только на скорости сходимости, но и на метриках качества итоговой модели. Будем рассматривать, исходя из условия задачи по выявлению токсичных комментариев (положительного класса, следующие статистики:

- сходимость loss функции (значение функции эмпирического риска)
- поведение accuracy на тестовой выборке
- precision и recall положительного класса на тесте
- F1-score как гармоническое среднее от precision/recall

5.1 Исследование learning-rate

В исследовании использовался метод инвертированной прогрессии для реализации **learning-rate**, формула которой представима в виде $\eta_i = \frac{\alpha}{\beta^i}$, где коэффициенты α и β являются гиперпараметрами. Такое определение последовательности достаточно удобно в работе, так как позволяет выбрать не только масштаб шага, но и скорость его убывания, при этом не требуя заранее известного числа итераций, как некоторые остальные методы задания **learning-rate**. В дальнейшем будем работать с моделями, имеющими L_2 регуляризацию с коэффициентом, равным 1.0, и не имеющими критерия остановки по разнице функции эмпирического риска на итерациях.

5.1.1 Сходимость модели

Первая характеристика каждой модели является ее скорость сходимости на предоставленной обучающей выборке. Оценим поведение сходимости на фиксированных данных для параметров α, β по сетке, выпустив "пучок" моделей с одинаковыми начальными данными для разных **learning-rate**.

Выбор beta состоит из набора $\{0, 0.3, 0.7\}$, так как он хорошо приближает реальную действительность. Если мы не хотим никак уменьшать шаг на итерациях, мы будем использовать 0, если же мы проваливаемся в локальные минимумы или не можем сойтись при большом количестве итераций к глобальному минимуму, однако не хотим испытывать проблемы с осцилляцией модели, будем брать малые значения, например 0.3, если нам надо очень быстро убрать шаг, будем использовать значения в области единицы, в нашем случае 0.7. При этом значения коэффициента масштаба α проходит значения множество значений от 10^{-3} до $10^{0.3}$

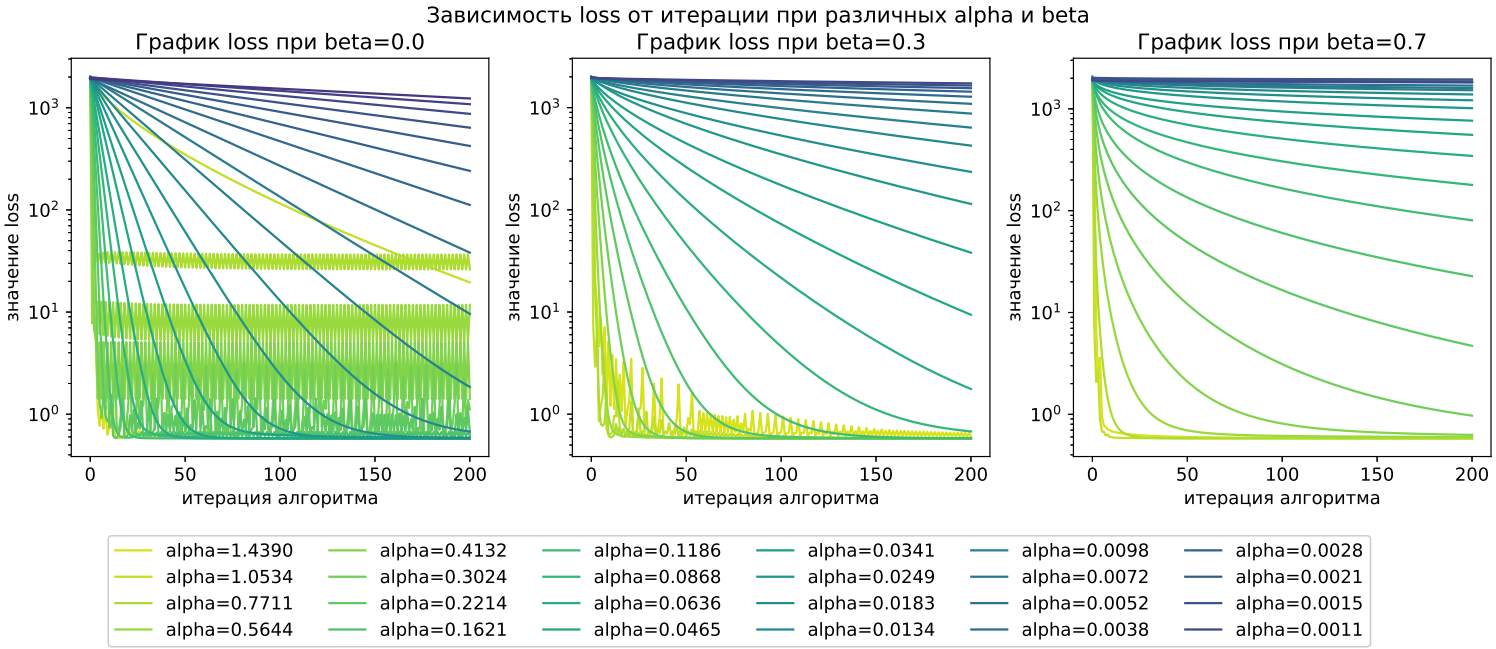

Figure 2: Сходимость алгоритмов GDC при различных параметрах **learning-rate**

Процесс обучения моделей с описанными коэффициентами изображен на графике 2. Цветовая гамма выбрана из соображений скорости сходимости моделей, и идет от синей гаммы — самых медленных и инертных моделей, к более быстрым, зеленым моделям. Можно легко заметить, что при любых β быстрее всего сходимость к плато присутствует у моделей с самым большим α, при чем с ростом коэффициента резко растет именно первые шаги, проходимые алгоритмом, что косвенно подтверждает предположение о природе и роли α. От β же явным образом зависит скорость убывания обучения, что наблюдается при сравнении трех графиков одновременно: с ростом β кривые становятся более пологими, а количество моделей, сумевших достичь оптимума за конечное число итераций, равное 200, с увеличением β монотонно уменьшается. Работу различных параметров можно сравнить с гибкостью обучения модели:

- параметр α — отвечает за гибкость модели на всем протяжении обучения.
- параметр β — отвечает за потерю такого качества, как гибкость, со временем

Отдельно в процессе обучения моделей присутствует особенность, сводящаяся на нет с ростом β: алгоритм, при относительно больших α и относительно малом β имеет свойство осцилляции в районе оптимума (ярко видно на примере β=0 и так же присутствует на β=0.3). Такое поведение говорит о неустойчивости процесса обучения, а потому его стоит избегать, хотя желание увеличить α естественно. Потому при росте α следует увеличивать и параметр β

5.1.2 Качество модели

Выберем семейство моделей с параметром $\beta = 0.3$, как некоторое усреднение полученных ранее результатов. Как предполагает задание, неотъемлемой частью итоговой модели должно являться умение определять положительный класс, делать это с хорошей точностью, не классифицируя обычную речь как токсичную, но и не пропускать действительно токсичные высказывания. Первое выливается в требования на достаточно больший *precision*, в то время как последнее выливается в требование на большой *recall*. Оценим поведение данных метрик на валидации для оговоренных моделей с помощью графиков 3

График precision показывает, как модели с относительно большим коэффициентом α имеют склонность к осцилляции около в плато, на которое они вышли. При этом модели с малыми коэффициентами демонстрируют противоположный результат, где они оказались не способными покинуть свое начальное (инициализированное) состояния ввиду своей малой гибкости на обучении.

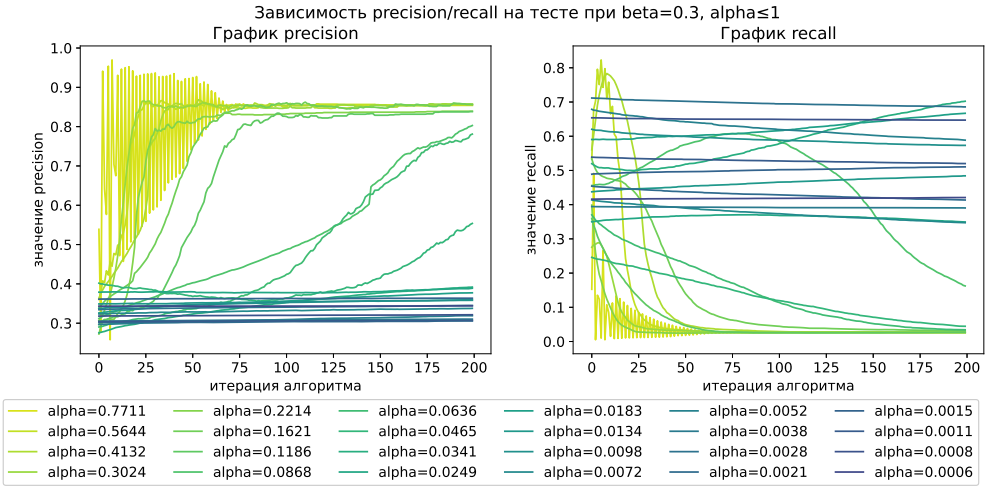

Figure 3: Precision и Recall моделей

При этом, если график precision радует хорошими показателями, график recall показывает крайне плохие результаты, относительно precision. Мы видим, как большинство моделей, с большими показателями α показывают около нулевые метрики recall, косвенно говоря, что модели предпочитают определению обоих классов константное предсказание, при котором почти всем объектам ставится в соответствие отрицательный класс.

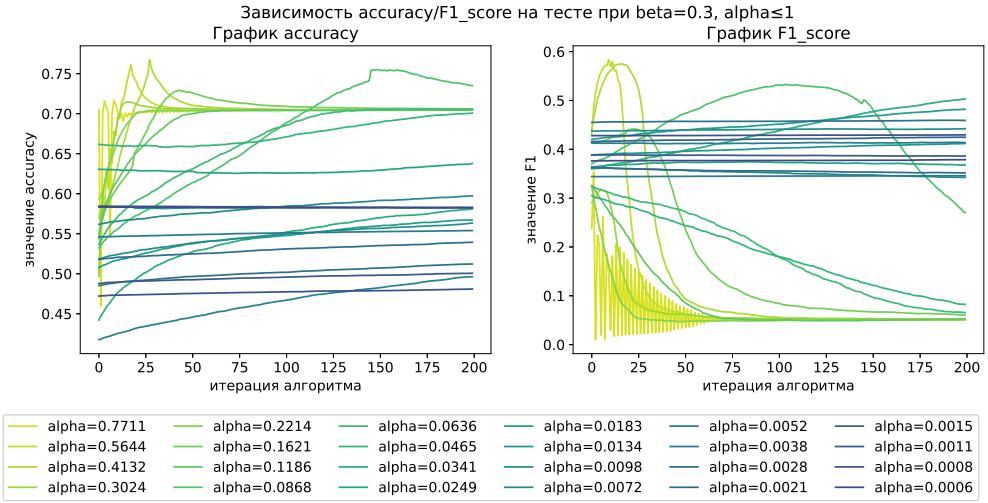

Figure 4: accuracy и F1-score моделей

Проясняет ситуацию график 4, где видно, что модели с минимальным recall быстро выходят на плато и остаются там на протяжении всех итераций обучения. Это ожидаемо, так как вся ошибка, влияющая на accuracy, заключена в тех положительных объектах, что были классифицированы как отрицательные. В тестовом датасете отношение числа объектов с отрицательным классом к положительным действительно равняется 7/3. Подобно precision, модели с малыми показателями α, остаются в начальном состоянии и не способны никак обучиться на данных.

График F1-score показывает усреднение гармоническим от precision и recall на каждой итерации. Видно, что самыми лучшими результатами обладают как раз необученные ввиду отсутствия гибкости (с малыми α) модели, оставшиеся по своей классификационной способности на уровне начального приближения. При этом, гибкие модели имеют крайне малый скор, что было ожидаемо из графика recall.

5.1.3 Почему такое могло произойти

Поведение модели явным образом указывает на то, что она посчитала предсказание отрицательного класса на всех объектах более выгодным, нежели другие стратегии. Такое поведение может

быть вызвано переобучением модели, либо несбалансированностью примеров классов в обучающей выборке. Для дальнейшей работы останемся с тем же датасетом.

5.2 Исследование начального приближения

Большую роль в сходимости и даже конечном результате может играть именно начальное приближение весов при инициализации модели. Именно от этого аспекта сильно зависит вопрос о попадании модели в локальные минимумы, время достижения глобального оптимума, то, какие показатели по метрикам будет показывать модель во время обучения.

Используются различные эвристики, что можно условно разделить на стохастические и детерминированные приближения. В исследовании рассматривались несколько способов определения начального приближения весов:

1. Приближение коэффициентами из нормального распределения
2. Приближение коэффициентами из равномерного распределения
3. Приближение коэффициентами из распределения Лапласа
4. Приближение коэффициентами из распределения Гампбела
5. Приближение нулевым вектором
6. Приближение значимостью признаков ($w_i = \frac{(f_i,y)}{(f_i,f_i)}$, где f_i — соответсвующий столбец признака у объектов)

> Если первые три распределения практически очевидны для понимания, то распределение Гампбела позволяет описать случай, большинство признаков объектов играет достаточно малую роль в определении положительного класса, при этом малое количество признаков играют сильную роль в определении отрицательного класса. Такая характеристика могла бы помочь в данной задаче.

Так как в опытах присутствует важный аспект случайности, необходимым является сбор статистической информации. За основу была взята модель с показателями **learning-rate** как 0.5, 0.5, с коэффициентом регуляризации L_2 равным 1.0. Для каждого из методов проведено 5 опытов и на их основе построены графики 5

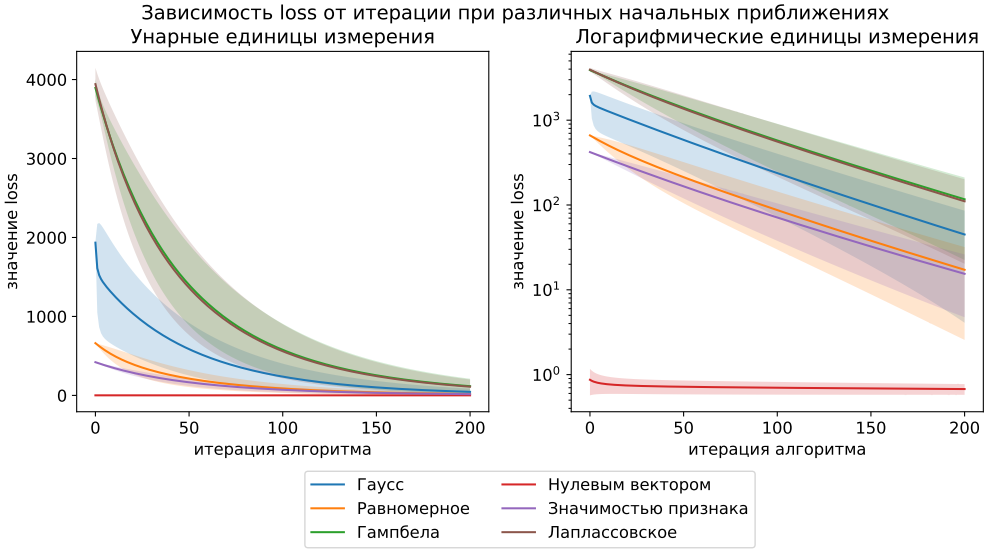

Figure 5: средний loss со среднеквадратическим отклонением при разных приближения

Видно, что большинство инициализирующих начальных приближений ведут себя примерно одинаково, имея большую ошибку при первых итерациях и быстро сходясь к минимуму при обучении. При этом среднеквадратическое отклонение значения функции эмпирического риска с ростом числа пройденных итераций обучения только увеличивается, что подтверждает начальное предположение о возможном попадании в локальные минимумы. Лучшим начальным

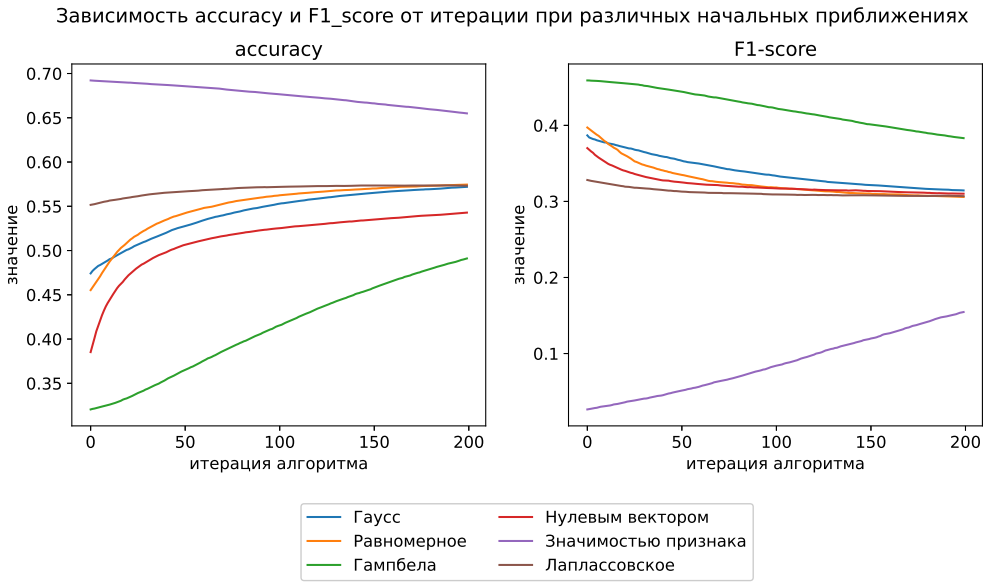

Figure 6: усредненные accuracy и F1-score для различных начальных приближений

приближением в этом плане оказалось приближением нулевым вектором, имея наименьшую дисперсию и наилучший loss начиная с первой же итерации.

Такое поведение приближения нулевым вектором объяснимо прошлыми наблюдениями, что предсказывание класса моделью, практически константно для всех объектов. Однако при этом по метрике F1-score 6 лидирует именно приближение Гампбела, что согласуется с выдвинутой теорией о применимости такого распределения. Большинство моделей стремятся к общему плато и ведут себя примерно одинаково. В отличие от приближения значимостью признака, активизирующего accuracy минимизацией F1-score.

6 Исследование стохастического градиентного спуска Логистической регрессии

Результаты градиентного спуска на первично обработанных данных не показал больших успехов. Однако оценим работу модели логистической регрессии, решающей оптимизационную задачу благодаря алгоритму стохастического градиентного спуска. Его главное отличие от алгоритма градиентного спуска заключается в том, что на каждой итерации модель высчитывает градиент шага только по подвыборке всех элементов. Такой подход способен быть более быстрым при подсчете и даже при обучении, а так же способен быть реализован таким образом, что в каждый момент времени компьютеру не обязан держать всю выборку в памяти. Однако при этом он более склонен к переобучению и сильно зависит от случайности.

Будем говорить, что алгоритм прошел "эпоху", когда он перебрал все возможные непересекающиеся подвыборки (почти одинаковой мощности) из датасета за цикл. Таким образом обучение представляет собой прохождение по частям датасета, а число итераций алгоритма равно числу эпох, пройденных им за время обучения.

Из определения стохастического градиентного спуска вытекает существование еще одного важного аспекта, от которого может зависеть обучение и поведение модели, помимо **learning-rate** и начального приближения - размер подвыборки, по которой алгоритм высчитывает градиент на итерациях. Изучим поведение стохастического градиента для задачи Логистической регрессии по этим трем аспектам.

6.1 Изучение learning-rate

Рассмотрим функцию эмпирического риска для рассматриваемой задачи с L_2 регуляризацией:

$$L(X^l, y; w) = \frac{1}{l}\sum_{i=1}^{l} \ln(1 + exp\{-y_i \langle w, x_i\rangle - w_0\}) + \frac{\gamma}{2}||w||^2$$

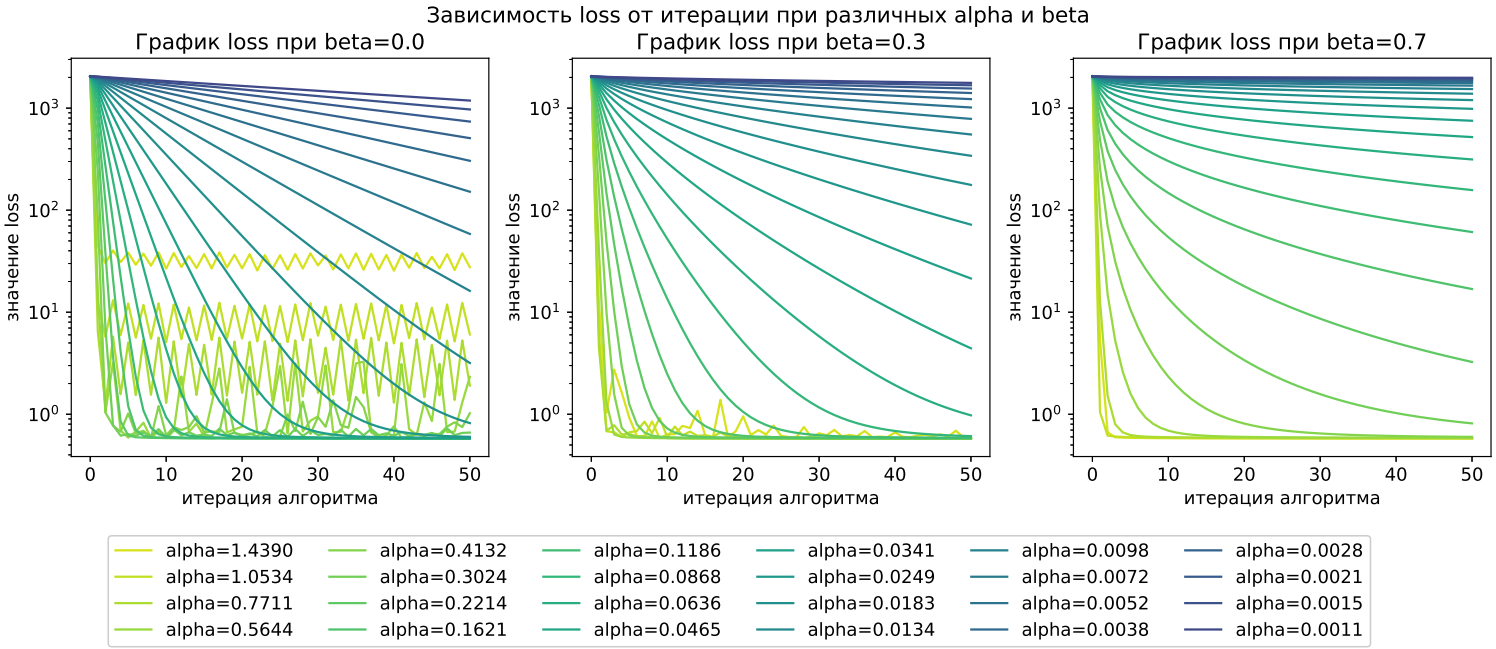

Figure 7: Сходимость алгоритмов SGDC при различных параметрах **learning-rate**

Данный функционал представляет собой сумму выпуклых функций (следует из определения логарифма) и строго выпуклой функции (добавочного члена регуляризации), из чего следует строгая выпуклость функции риска. Из курса методов оптимизации, для такого функционала действует теоремы "О выпуклой оптимизации" и "О стохастической аппроксимации", говорящие, что при правильно выборе скорости сходимости (**learning-rate**) градиентный спуск сходится почти наверное к глобальному минимуму. Таким образом начальное предположение состоит в том, что стохастический градиентный спуск будет вести себя примерно так же на данных, что и градиентный спуск. Будем собирать статистику по модели за время обучения дважды за одну эпоху. Число эпох поставим равным 200.

6.1.1 Обучение

Таким же образом, что и для градиентного спуска, выпустим "пучок" моделей по оговоренной сетке с начальным приближением вектором из нормального распределения. Графики динамики обучения представлены ниже 16. Как можно заметить, главное различие между графиками для **GDC** (градиентного спуска) и **SGDC** (стохастического градиентного спуска) является большая ломаность кривых обучения, вызванное тем, что алгоритм раз за разом переобучается на малых наборах объектов. Однако общая зависимость обучения от параметров α и β повторяет таковую для градиентного спуска.

6.1.2 Качество алгоритмов

Сравним статистики accuracy и F1-score относительно положительного класса, как основополагающие оценки при анализе данной задачи. Сразу заметен тот факт, что некоторым алгоритмам удалось улучшить свое качество, относительно алгоритма градиентного спуска. За 200 итераций некоторые алгоритмы достигли качества в 0.75, при максимуме в 0.71 для GDC. При этом здесь же в большей степени проявляется большая ломанность кривых, что была отмечена еще на графиках функции эмпирического риска, вызванная обучаемостью алгоритма каждый раз на отдельных пачках объектов.

Так же заметно большее число алгоритмов, сумевших достичь относительно большего показателя F1-score при большом показателе accuracy. Данное наблюдение возможно из того, факта, что для стохастического градиентного спуска несбалансированность классов играет меньшую роль, нежели чем для градиентного спуска.

6.2 Изучение начальных приближений

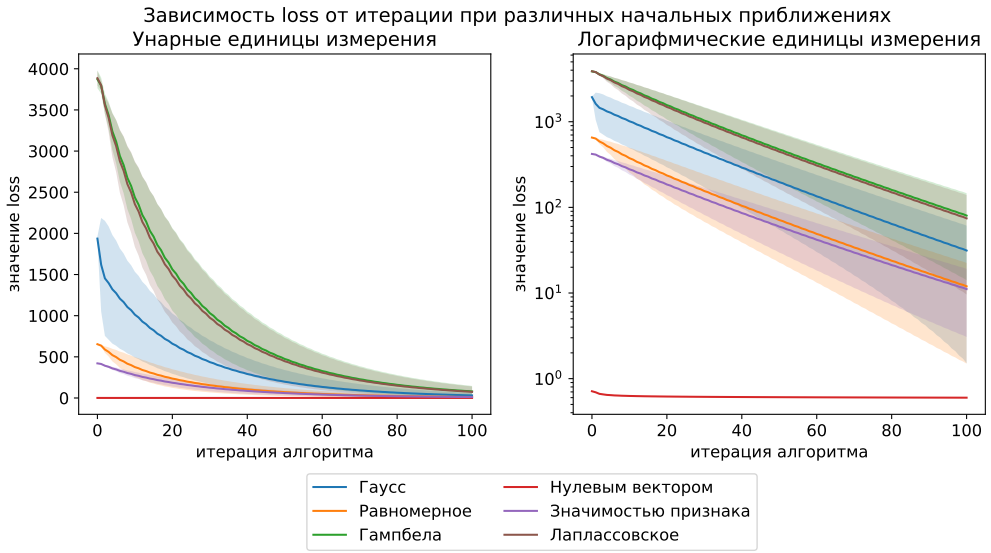

Figure 9: усредненный loss при обучении для различных начальных приближений SGDC

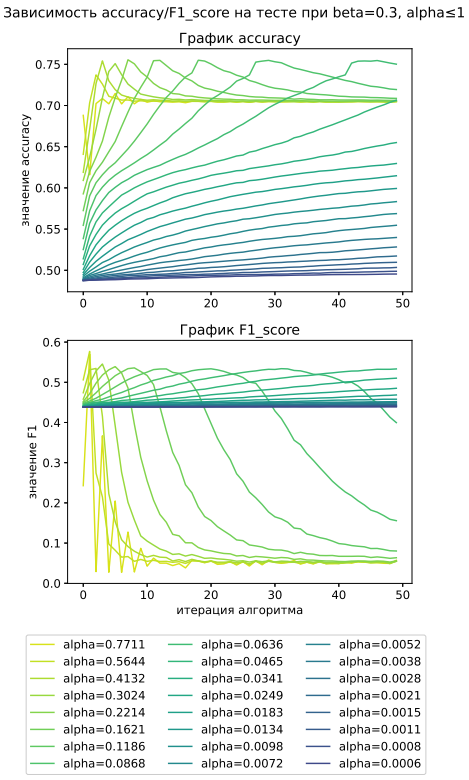

Figure 8: График изменения метрик accuracy и F1 на тесте в процессе обучения

Посмотрим на процесс обучения моделей с коэффициентом регуляризации равным 1 и значения α и β по 0.5 на графике 9. Видно, что стохастический градиентный спуск даже смог улучшить картину обучения, для большинства приближений уменьшив дисперсию значений. Однако в общих чертах поведение по сходимости моделей остается неизменным для обоих оптимизационных алгоритмов.

Что является более важной характеристикой, так это конечное качество алгоритмов, представленное на графике 10. Мы можем видеть общее улучшение качества по метрике F1-score для практически всех начальных приближений. Однако наилучшим образом проявило себя приближение значимостью признака, в отличие от градиентного спуска. Фиолетовая кривая показывает большие показатели как precision, так и recall, имея при этом большой коэффициент accuracy, а факт того, что данный график является усреднением от нескольких опытов, показывает эффективность именно такого подхода к построению модели на данных. Так же стоит отметить второй по характеристикам метод приближения распределением Гампбелла.

6.3 Оценка зависимости обучения SGDC от размера подвыборок

Алгоритм стохастического градиента, как было замечено, в каждый момент времени работает лишь с частью всей обучающей выборки. Вводится понятие **batch-size**, размер таких групп, либо как абсолютное число объектов, попадающий в каждую группу, либо как доля объектов из доступной выборки, что на каждой итерации попадает в свою группу. В общем случае, на каждой эпохе объекты перемешиваются, чтобы убрать эффект переобучения.

Зафиксируем модель с коэффициентом регуляризации L_2 равным 1.0 и коэффициентами α и β равными 0.5. Модель обучалась на фиксированной обучающей выборке с различными коэффициентами **batch-size** в диапозоне от 0.05 до 0.65. Далее модель считается обученной, если при итерации loss изменяется менее чем на 10^{-5}.

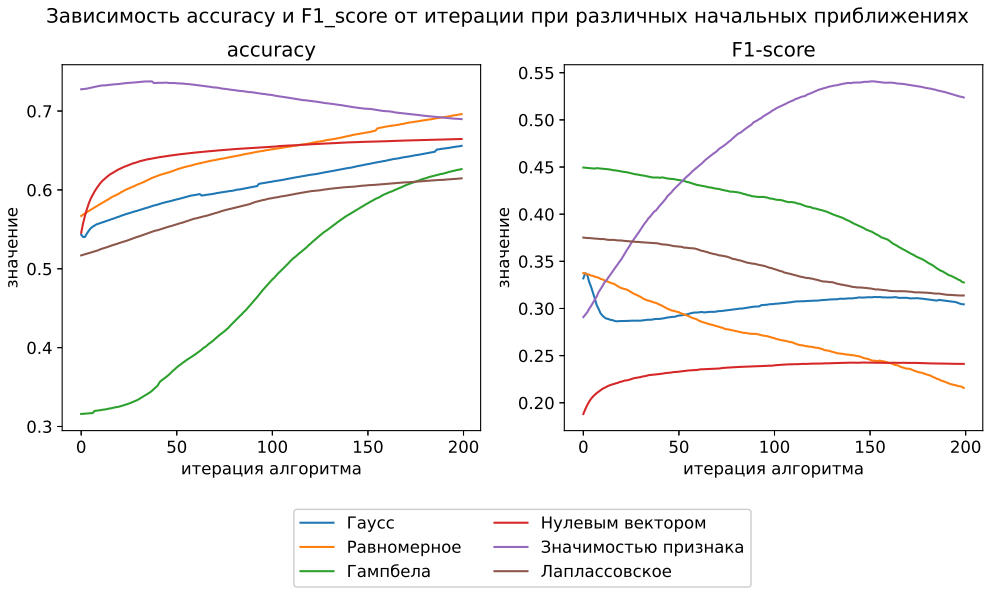

Figure 10: accuracy и F1-score моделей для разных начальных приближений

На графике 11 полученная зависимость времени обучения от коэффициента, характеризующего **batch-size**. При уменьшении такого показателя время обучения в среднем падает, непосредственно показывая, что модели намного легче подстраиваться под каждый объект по отдельности, нежели делать это для подвыборок больших размеров.

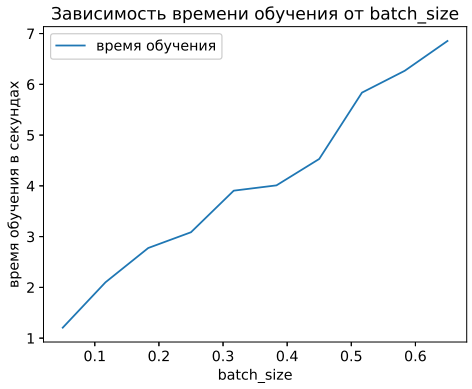

Figure 11: Общая зависимость времени обучения от **batch-size**

7 Сравнение GDC с SGDC

Стохастический градиентный спуск имеет ряд преимуществ по сравнению с детерминимированной версией, что в зависимости от некой удачи могут сильно улучшить характер работы конечной модели. Все нужные выводы были сделаны ранее, а потому краткая компиляция:

- Градиентный спуск позволяет оценивать задачу оптимизации для всех объектов в целом. Если для числа задач такой подход может быть намного более удобным, даже необходимым, то в случае данной задачи, такое поведение оптимизатора способно только ухудшить положение при неправильно подготовленных данных.
- Стохастический градиентный спуск напротив, способен решить проблему несбалансированности выборки, убрать переобученность алгоритма при осознанном ограничении числа эпох обучения.
- Стохастический алгоритм обучает модель менее "устойчивым" путем
- Подход стохастического градиентного спуска позволяет организовать процесс обучения более гибким к памяти компьютера образом, не вынуждая хранить всю выборку в ОП постоянно.
- Поведение алгоритмов относительно изменения **learning-rate** мало отличимо и разница нивелируется при увеличении количества итераций.
- Стохастический градиентный спуск позволяет проводить обучение модели намного быстрее детерминированного.

Можно говорить, что в рамках данной задачи стохастический градиент показал себя более успешно по времени обучения и по конечным метрикам качества. Потому дальнейшую работу предполагается производить на основе **SGDC**. Однако даже так нельзя утверждать ничего в целом о действительной работоспособности этих моделей относительно друг друга, так как немалую роль в успехах и неуспехах играла именно обучающая выборка, или иначе, ее предобработка. Именно анализ и возможных улучшений в обработке текстовой информации способен привнести большее качество в модели классификации.

8 Углубленное построение пространства признаков

Как отмечалось ранее, специфика поставленной задачи заключается в отсутствии первоначального пространства признаков у объектов. Его поиск является первым и самым важным этапом на пути решения проблемы в целом (классическая задача **Feature-Selection**). Однако в отличие от, например, прямого построения моделей на подготовленных данных, здесь нельзя напрямую отталкиваться от формализованных метрик качества, а сама задача выливается в набор различных эвристик, которые перебираются в поиске наилучшей.

Выбранной и фиксированной моделью представления данных является Мешок слов. Однако он, как отмечалось ранее, достаточно сильно зависит как от предобработки текстов, так и от установленного отображения элементов строк в пространство признаков. Задача явным образом делима на две составляющих, а потому рассмотрим каждый из аспектов.

8.1 Предобработка текстов

Вне зависимости от методов определения оператора отображения в пространство численных признаков ($\mathcal{V}_{cv}, \mathcal{V}_{fidv}$), вопрос самих отображаемых данных стоит на первом месте. Без качественной обработки символьных цепочек (текстов) на первичном уровне, возникают проблемы с информативностью отдельных лексем, что затрагивались ранее в отчете. Потому рассматривались два основных подхода очистки и приведения текстов с последующим их сравнением с уже используемыми, практически нетронутыми данными: метод **лемматизации** и **стемминга**. Воздействуем на все три выборки стандартным $\mathcal{V}_{cv}$, отбрасывающим все слова, встречающиеся реже, чем на 10 текстах.

Полученные признаковые пространства имеют размерности 11222, 7874 и 9816 для строк обработанных первоначальными преобразованиями, для данных после стемминга и после лемматизации соответственно. Видно, что наибольшая размерность присуще именно практически нетронутым данным, что вполне ожидаемо. Методы лемматизации и стемминга достаточно сильно уменьшают данное значение, при чем стемминг делает это более агрессивно (практически в два раза относительно лемматизации). Последний факт может как сильно улучшить поведение модели, ее качество и скорость работы, так и значительно ухудшить. Сравним выборки на основе скорости обучения моделей в среднем и итогового качества.

8.1.1 Скорость сходимости

Будем обучать модель с **learning-rate** равным инвертированной прогрессии с параметрами из сетки: $\beta \in [0.01; 1.0], \alpha \in [0.1, 12]$, приняв коэффициент регуляризации L_2 равным 0 и размер батчей для **SGDC** равным 0.25 от всей выборки.

Усреднив время обучения по β, отобразим время затраченное алгоритмами для каждого α, высчитывая его как дельта разница между временем начала обучения и временем окончания последней итерации. Итоговые показатели отображены на графике 12. Относительно α заметна общая тенденция на уменьшение времени обучения с ростом коэффициента, что является ожидаемым эффектом и наиболее выражено у данных после применения стемминга. Однако имея одинаково плохую сходимость в точке минимального α, модели, обученные на данных с различной примененной обработкой, ведут себя по-разному при изменении коэффициента. Если стемминг с ростом α показывает стабильное повышение скорости обучения, то лемматизированные данные сильно зависят от коэффициента.

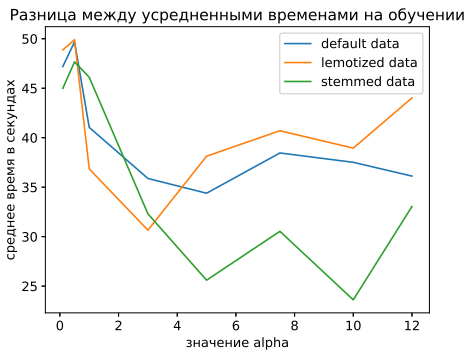

Figure 12: Общая зависимость времени обучения от **batch-size**

Для лемматизированных данных время, затраченное на обучения, стремительно падает до определенного момента и вновь повышается на уровень с изначальными данными. Такое поведение объясняется тем, что лемматизированные данные, в отличие от стеммированных, сохраняют больше "контекста" слова, а потому больший **learning-rate** не позволяет им сойтись, заставляя модель осциллировать около оптимума.

8.1.2 Качество получаемых моделей

Оценка качества обучения производится на основе конечных показателей метрик качеств моделей: на основе accuracy и F1-score. Данные о тестах агрегированы в таблицы tables 4 to 6, где значения метрик на валидационой выборке приведены для всех комбинаций коэффициентов **learning-rate**.

Видно, что на каждой из выборок модели смогли достигнуть примерно одинакового наилучшего результата: accuracy 0.89, F1-score 0.827, при чем такие результаты были достигнуты при самых мылых коэффициентах β, что можно связать с большой размерностью эмбеддингового пространства. При этом для каждого метода существует своя α, доставляющая лучшее качество на валидации, возле которой кучкуются другие наилучшие результаты модели по данной выборке. Тем самым косвенно показывается, что соответственный выбор коэффициентов находится в области устойчивого оптимума выбора **learning-rate**.

В общем случае заметна большая разница между результатами, полученными ранее (при анализе градиентного спуска) и здесь. Видно, что предобарботка документов способна крайне сильно улучшить классификационную способность алгоритма. Вне зависимости от того, что модель Логистической регрессии смогла показать свою способность доставлять хорошее качество на каждой из доступных выборок, заметно общее превалирование по метрикам качества моделей, обученных на выборках с лемматизацией и стеммингом относительно моделей, обучавшихся на стандартных данных.

8.2 Исследование оператора перевода $\mathcal{V}$

Качественная предобработка текстовой информации при прочих равных, способна улучшить качество модели, уменьшить время обучения, а так же изменить размерность итогового признакового пространства. Однако главная роль в создании итоговых эмбеддингов отводится именно оператору перевода. В зависимости от выбора стратегии (учета или неучета частоты встречи слов) и входных параметров min_{df} и max_{df} (минимального и максимального числа встреч слова соответственно, чтобы попасть в итоговое пространство), устройство эмбеддингов может кардинально отличаться для двух разных операторов, что несомненно отразится и на работе модели. Проведем оценку такого влияния каждого из двух операторов на выборку с лемматизацией, и сравним их между собой.

Будем прогонять по сетке значения min_{df} и max_{df} для двух операторов, взяв за основу мер качества три критерия:

- Размерность итогового признакового пространства — меньшее признаковое пространство позволяет избежать многих проблем с переобучением модели, свести требуемый размер обучающей выборки к минимуму, придать модели больше устойчивости во время обучения
- Время обучения модели
- Качество получаемых моделей

Пройдемся по порядку по каждому из пунктов.

8.2.1 Размерность пространства признаков

Так как операторы $\mathcal{V}_{cv}$ и $\mathcal{V}_{fidv}$ определяют размерность пространства единым образом, различаясь лишь в вопросе определения компонент вектора эмбеддинга для объектов, будем рассматривать зависимость размерности для двух операторов одновременно, в зависимости от min_{df}, max_{df} и изначальной выборки.

Table 1: Размерность признаков

min\max	40	75	150	250	600	1
0	75474	76832	77875	78415	78948	79465
10	5825	7183	8226	8766	9299	9816
20	2342	3700	4743	5283	5816	6333
50	-1	881	1924	2464	2997	3514
100	-1	-1	575	1115	1648	2165
200	-1	-1	-1	227	760	1277
300	-1	-1	-1	-1	392	909
500	-1	-1	-1	-1	96	613

Видно, как при малейшем увеличении нижней границы (с нуля до 10) по допуску слов, стремительно падает размерность признакового пространства. Это следует из присутствия в данных множественных примеров лексем, почти более не встречаемых в корпусе текста. Классификация по таким словам будет работать крайне плохо, так как они с малой вероятностью являются типичными представителями своего класса, а потому могут быть нами проинтерпретированы в качестве выбросов. При дальнейшем увеличении нижней границы убавление размерности наблюдается, но скорость сильно уменьшается. Такое поведение может отражать ситуацию, когда часть отсекаемых слов все же несла смысловую нагрузку. При этом видно, что существуют и противоположные слова (порядка 500 штук) что встречаются в большом числе текстов (больше 600). Такие слова могут быть так же не эффективны при обучении модели, так как являются скорее типичными для данного корпуса текстов, нежели типичными отдельному классу.

8.2.2 Оценка времени обучения

Время обучения, как мера качества алгоритма, может не быть лишь показателем быстродействия модели и ее удобством использования. Время, затраченное на обучение, показывает скорость сходимости модели, а потому посредством сравнения полученных результатов может многое сказать об эффективности обучения модели на данных. Все так же обучим набор моделей **SGDC**, с коэффициентом регуляризации $L_2 = 0.01$, и равными для всех коэффициентами **learning-rate**: $\alpha = 7, \beta = 0.01$.

Table 2: Сравнение времен обучения моделей после применения операторов перевода в секундах

min\max	Время при обработке CountVectorize						Время при обработке TfidfVectorizer					
	40	75	150	250	600	1	40	75	150	250	600	1
0.0	0.969	0.931	2.573	6.841	2.615	9.949	2.045	2.286	2.624	2.752	3.986	11.842
10	0.323	0.43	0.633	1.842	0.651	14.473	1.221	3.035	1.816	2.988	2.867	8.108
20	0.576	0.398	0.265	2.637	2.407	10.944	1.034	2.555	1.669	2.03	2.712	9.214
50	-1	0.042	0.362	0.727	0.223	11.685	-1	0.985	1.362	1.716	2.448	10.382
100	-1	-1	0.266	0.6	0.316	11.744	-1	-1	2.156	1.514	2.222	13.21
200	-1	-1	-1	0.398	0.086	7.862	-1	-1	-1	1.815	1.711	7.208
300	-1	-1	-1	-1	0.047	7.036	-1	-1	-1	-1	1.377	6.592
500	-1	-1	-1	-1	0.022	6.408	-1	-1	-1	-1	0.984	6.136

Из таблицы 2 видно, что на большинстве значений пар min_{df} и max_{df} метод **CountVectorizer** сходится быстрее при прочих равных, в редких случаях отставая от **TfidfVectorize**, что можно считать погрешностью работы компьютера. Сопоставляя с данной таблицей таблицу с размерностями пространств 1, можно увидеть, что при уменьшении размерности пространства модели, обучаемые на данных, обработанных **CountVectorizer**, сходятся на порядок быстрее за исключение случая при $max_{df} = 1.0$.

8.2.3 Сравнение качества моделей

Сравним классификационную способность моделей, обученных на каждой из выборок, полученных после применения соответствующих операторов перевода. Вопреки желанию зафиксировать отдельные модели и прогнать каждую на полученных датасетах для каждого $\mathcal{V}$, в общем случае так делать неверно. При прочих равных, оператор перевода $\mathcal{V}$ достаточно сильно воздействует на сходимость. Зафиксируем из соображений охвата наибольшего числа возможных вариантов выбора min_{df} и max_{df} несколько пар, для каждой из которых определим оператор перевода и посмотрим на динамику обучения моделей на соответствующих получившихся выборках.

Выбрали несколько выделяющихся вариантов для пар параметров:

- $min_{df} = 10, max_{df} = 1.0$ - выбор такого оператора может быть обусловлен желанием сохранить как можно больше возможных полезных лексем, в предположении, что от возможно затесавшихся вместе с ними шумовых слов будет меньше вреда, чем от интересующих нас слов пользы.

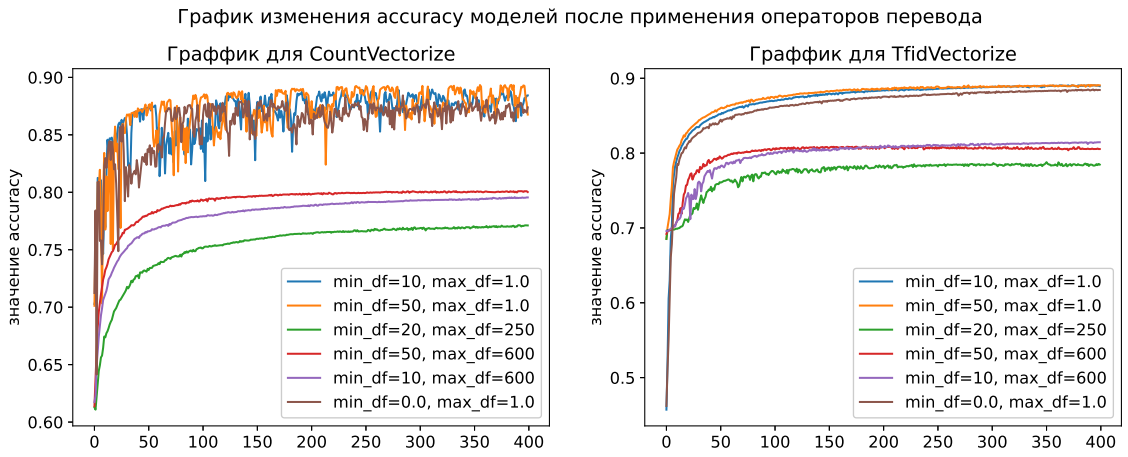

Figure 13: Сравнение динамики изменения accuracy на валидации при применении $\mathcal{V}_{cv}$ и $\mathcal{V}_{fidv}$

- $min_{df} = 50$, $max_{df} = 1.0$ - более агрессивный вариант, при котором мы хотим отсечь большинство (если не все) шумовые слова. Стоит быть осторожным, так как из таблицы 1 следует, что при нижнем пороге отсечения в 50 встреч в текстах вполне могут отбрасываться значащие (полезные для классификации) слова.
- $min_{df} = 20$, $max_{df} = 250$, $min_{df} = 10$, $max_{df} = 600$, $min_{df} = 50$, $max_{df} = 600$ - специальные "внутренние" варианты выбора рамок для слов. Выбор таких пределов отбрасывания может быть вызван желанием оставить лишь самые эффективные, часто встречаемые, но при этом не шумовые слова, заодно существенно понизив размерность пространства.
- $min_{df} = 0.0$, $max_{df} 1.0$ - специальный случай, при котором мы оставляем все слова в выборке.

Перебрав по сетке параметры для α и β, приняв размер батча для стохастического градиентного спуска за долю в 0.1 от всей выборки, нашли наилучшие коэффициенты, что привели к наибольшим показателям для моделей: $12, 0.2$ для CountVectorize и $13, 0.01$ для TfidVectorize соответственно. Получившиеся результаты отображены на графиках 13, из которых вытекает несколько важных наблюдений:

1. Модели, построенные на данных от $\mathcal{V}_{cv}$ и $\mathcal{V}_{fidv}$ по своей асимптотике ведут себя крайне схоже, выходя на одни и те же плато на примерно одной и той же итерации
2. Модели, построенные на основе данных, преобразованных $\mathcal{V}_{fidv}$, на обучении ведут себя намного стабильнее. Такое поведение говорит о большей предсказуемости модели и при прочих равных является крайне желанной особенностью.
3. Несмотря на начальные предположения об отсечении слишком часто встречаемых слов в текстах, именно модели, не имеющие верхней границы для слов показали лучшее качество
4. В рамках **данной конкретной** задачи отсечение большинства слов-шумов (около 60000 лексем) влияет не сколько на качество алгоритма, сколько на скорость сходимости, так как такие модели имеют метрики качества наравне с лучшими.

8.3 Итоги предобработки данных

Главной задачей всей предобработки является создание таких данных, на которых модель смогла бы обучаться наиболее предсказуемо и эффективно. Фактически, в классических задачах это достигается внутренними средствами моделей, однако специфика данной задачи в том, что сами данные нужно получать самостоятельно. Очевидным кажется тот факт, что применение средств первичной обработки документов вместе с правильным подбором оператора перевода вместе способны кардинально улучшить работу конечной модели.

> В дальнейшем будем использовать обработку лемматизацией с последующим применение TfidVectorizer с параметрами $min_{df} = 50$, $max_{df} = 1.0$.

9 Реализация наилучшей модели

Соберем наилучшую модель на основе данных, полученных ранее, и проанализируем результаты. Для этого обработаем начальные данные с помощью алгоритма лемматизации с отсеканием слов частотой встреч до 50 в корпусе текстов ($min_{df} = 50$, $max_{df} = 1.0$) и применением алгоритма TfidfVectorizer; на их основе построим модели **SGDC** с параметрами размера батчей в 0.1 от всей обучающей выборки и прогоним параметры **learning-rate** по сетке, выбрав итоговую модель, с наилучшим accuracy 3

upper\lower	8	9	10	11	12	13	14	15	16	17	18
0.01	0.888	0.888	0.88	0.89	0.89	0.891	0.891	0.89	0.889	0.89	0.891
0.05	0.885	0.887	0.882	0.888	0.888	0.889	0.889	0.89	0.875	0.891	0.891
0.2	0.864	0.868	0.87	0.872	0.873	0.875	0.876	0.878	0.879	0.88	0.881
0.6	0.759	0.765	0.772	0.777	0.782	0.786	0.789	0.793	0.796	0.798	0.801

Table 3: Итоговое качество на моделях при поиске самой наилучшей

Большинство моделей имеют примерно хорошие результаты точности (accuracy на валидации) по прошествии обучения, косвенно подтверждая, что проблема с нахождением подходящего обработчика текстовой информации решена успешно. Выберем модель с показателями $\alpha = 18$, $\beta = 0.05$, как оптимум, окруженный при этом прочими моделями, достигшими точно такого же результата.

Посмотрим на характеристики классификационной способности алгоритма более внимательной. На графиках 14 изображены графики Precision/Recall относительно положительного класса. При чем мы можем видеть достаточно хорошие результаты, так например **ROC-AUC** составляет 0.9545, а **PR-AUC** 0.9056, что говорит большой полноте и точности классификации положительного класса не только на данной моделей, но и на семействе моделей при изменении **treshhold** определения положительного класса.

Итоговый объект всей работы получен. Однако даже продвинувшись так далеко классификатор все еще имеет объекты, на которых допускает ошибки. Проведем анализ таких ошибочных объектов.

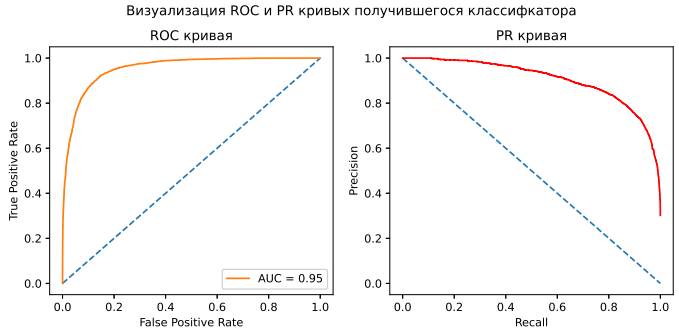

Figure 14: ROC и Precision/Recall кривые лучшего классификатора

9.1 Анализ ошибок классификатора

Построим матрицы ошибок, чтобы оценить, куда перевешивает модель в предсказаниях 15. Отобразим для удобства матрицу ошибок полученного классификатора, нормализованную относительно числа положительных предсказаний модели и числа положительных классов в валидационной выборке. Виден явный, хоть и не большой, перевес в сторону классификации отрицательного класса (здесь 0) как положительного (ошибка первого рода). Из семантики задачи пропуск именно такой ошибки приводит к наибольшему ущербу. Посмотрим на то, как выглядят ошибки классификатором.

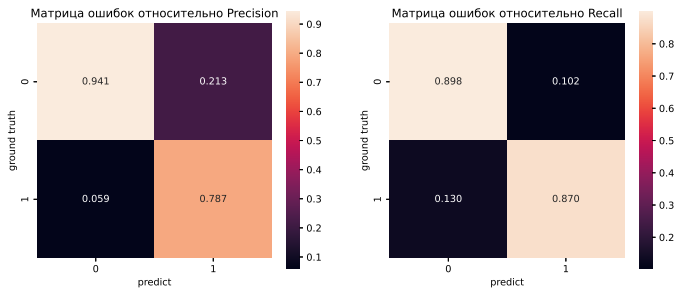

Figure 15: Нормализированные матрицы ошибок на валидации

Примеры:

- "== Gay? == He's gay too. It should be noted that he has a male partner."
- "Hello everyone I'm just here to tell you that you're all freaks"
- "I WILL BURN YOU TO HELL IF YOU REVOKE MY TALK PAGE ACCESS!!!!!!!!!!!!!"
- "666 the devil will get you all !!!!!!!!!!!!!!!!"
- "You wikipedia nazis need to stop removing information"

Как можно видеть, вместе с рядовыми ошибками классификации по пограничным, не всегда даже явно классифицируемых человеком высказываний есть явные ошибки разметки. Это одна из распространенных проблем в машинном обучении, особенно **NLP**. Бороться с данным типом ошибок практически не возможно в рассматриваемой модели, потому модель можно рассматривать как достаточно качественную.

9.2 Какие слова оказались самыми важными для классификации

Определить данный факт достаточно просто, благодаря рассматриваемой модели линейной регрессии, где абсолютную величину соответственного коэффициента можно трактовать как значимость соответственного признака. Таковыми являются слова: *fuck, fucking, shit, idiot* и так далее.

Figure 16: Облако главных 100 слов, имеющих вес в классификации. *Внимание, идея принадлежит не мне. Позаимствовал ее у своего одногруппника. Но она очень красивая, потому просто был обязан вставить.*

Table 4: Качество алгоритмов, обученных на данных с **первичной** обработкой

beta\alpha	0.1		0.5		1.0		3.0		5		7.5		10		12	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
0.01	0.756	0.583	0.813	0.69	0.796	0.663	0.865	0.784	0.884	0.807	0.889	0.818	0.876	0.813	0.821	0.76
0.05	0.746	0.568	0.804	0.674	0.806	0.681	0.839	0.748	0.874	0.791	0.866	0.788	0.885	0.819	0.839	0.777
0.1	0.734	0.549	0.794	0.655	0.818	0.7	0.821	0.703	0.772	0.665	0.854	0.784	0.877	0.809	0.883	0.807
0.2	0.705	0.502	0.77	0.61	0.796	0.659	0.809	0.686	0.804	0.67	0.845	0.748	0.842	0.76	0.871	0.785
0.4	0.656	0.431	0.725	0.533	0.753	0.579	0.793	0.653	0.81	0.686	0.816	0.697	0.785	0.645	0.726	0.608
0.7	0.602	0.342	0.656	0.431	0.686	0.476	0.732	0.547	0.753	0.579	0.767	0.604	0.778	0.624	0.785	0.638
1	0.582	0.304	0.61	0.358	0.635	0.397	0.678	0.463	0.698	0.492	0.716	0.519	0.729	0.542	0.737	0.554

Table 5: Качество алгоритмов, обученных на данных с обработкой **Лемматизацией**

beta\alpha	0.1		0.5		1.0		3.0		5		7.5		10		12	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
0.01	0.756	0.586	0.822	0.707	0.805	0.681	0.877	0.79	0.89	0.819	0.849	0.788	0.892	0.827	0.827	0.767
0.05	0.744	0.564	0.813	0.691	0.817	0.697	0.801	0.7	0.885	0.812	0.891	0.821	0.891	0.826	0.892	0.828
0.1	0.727	0.531	0.799	0.666	0.806	0.678	0.786	0.649	0.836	0.692	0.887	0.815	0.889	0.821	0.892	0.824
0.2	0.699	0.471	0.774	0.619	0.803	0.674	0.81	0.69	0.82	0.702	0.85	0.741	0.831	0.751	0.872	0.8
0.4	0.659	0.367	0.718	0.511	0.752	0.578	0.798	0.665	0.819	0.702	0.818	0.699	0.812	0.686	0.809	0.69
0.7	0.627	0.282	0.659	0.367	0.681	0.424	0.726	0.529	0.752	0.579	0.77	0.612	0.783	0.635	0.791	0.65
1	0.595	0.267	0.632	0.291	0.643	0.324	0.675	0.414	0.694	0.462	0.712	0.5	0.727	0.532	0.736	0.551

Table 6: Качество алгоритмов, обученных на данных с обработкой **Стеммингом**

beta\alpha	0.1		0.5		1.0		3.0		5		7.5		10		12	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
0.01	0.766	0.608	0.825	0.716	0.837	0.737	0.83	0.722	0.814	0.655	0.89	0.813	0.837	0.772	0.871	0.81
0.05	0.756	0.592	0.816	0.7	0.836	0.736	0.844	0.748	0.822	0.701	0.871	0.8	0.851	0.786	0.873	0.81
0.1	0.744	0.574	0.805	0.679	0.828	0.723	0.84	0.744	0.828	0.717	0.847	0.757	0.862	0.788	0.87	0.8
0.2	0.719	0.542	0.782	0.637	0.808	0.685	0.836	0.736	0.844	0.751	0.836	0.739	0.825	0.709	0.837	0.733
0.4	0.655	0.464	0.737	0.563	0.762	0.603	0.804	0.678	0.822	0.711	0.834	0.731	0.838	0.74	0.843	0.747
0.7	0.587	0.379	0.655	0.464	0.691	0.509	0.744	0.573	0.763	0.604	0.778	0.631	0.789	0.651	0.798	0.665
1	0.555	0.366	0.597	0.387	0.629	0.426	0.684	0.503	0.711	0.536	0.729	0.558	0.741	0.572	0.751	0.586